Predictive CDN Selection for Video Delivery Based on LSTM Network Performance Forecasts and Cost-Effective Trade-Offs

Roberto Viola, Angel Martin, Javier Morgade, *Senior Member, IEEE*, Stefano Masneri, Mikel Zorrilla, Pablo Angueira, *Senior Member, IEEE*, and Jon Montalbán, *Senior Member, IEEE*

***Abstract*—Owing to increasing consumption of video streams and demand for higher quality content and more advanced displays, future telecommunication networks are expected to outperform current networks in terms of key performance indicators (KPIs). Currently, content delivery networks (CDNs) are used to enhance media availability and delivery performance across the Internet in a cost-effective manner. The proliferation of CDN vendors and business models allows the content provider (CP) to use multiple CDN providers simultaneously. However, extreme concurrency dynamics can affect CDN capacity, causing performance degradation and outages, while overestimated demand affects costs. 5G standardization communities envision advanced network functions executing video analytics to enhance or boost media services. Network accelerators are required to enforce CDN resilience and efficient utilization of CDN assets. In this regard, this study investigates a cost-effective service to dynamically select the CDN for each session and video segment at the Media Server, without any modification to the video streaming pipeline being required. This service performs time series forecasts by employing a Long Short-Term Memory (LSTM) network to process real time measurements coming from connected video players. This service also ensures reliable and cost-effective content delivery through proactive selection of the CDN that fits with performance and business constraints. To this end, the proposed service predicts the number of players that can be served by each CDN at each time; then, it switches the required players between CDNs to keep the (Quality of Service) QoS rates or to reduce the CP's operational expenditure (OPEX). The proposed solution is evaluated by a real server, CDNs, and players and delivering dynamic adaptive streaming over HTTP (MPEG-DASH), where clients are notified to switch to another CDN through a standard MPEG-DASH media presentation description (MPD) update mechanism.**

***Index Terms*—Content delivery network, MPEG-DASH, operational expenditure, quality of service.**

Manuscript received May 26, 2020; revised September 18, 2020; accepted September 29, 2020. Date of publication November 3, 2020; date of current version March 4, 2021. This work was supported in part by the EC projects Fed4Fire+, under Grant 732638 (H2020-ICT-13-2016, Research and Innovation Action), and in part by Open-VERSO project (Red Cervera Program, Spanish Government's Centre for the Development of Industrial Technology). *(Corresponding author: Roberto Viola.)*

Roberto Viola is with the Vicomtech Foundation, Basque Research and Technology Alliance, 20009 San Sebastián, Spain, and also with the Department of Communications Engineering, University of the Basque Country (UPV/EHU), 48013 Bilbao, Spain (e-mail: rviola@vicomtech.org).

Angel Martin, Javier Morgade, Stefano Masneri, and Mikel Zorrilla are with Vicomtech Foundation, Basque Research and Technology Alliance, 20018 San Sebastián, Spain.

Pablo Angueira is with the Department of Communications Engineering, University of the Basque Country (UPV/EHU), 48013 Bilbao, Spain.

Jon Montalbán is with the Department of Electronic Technology, University of the Basque Country (UPV/EHU), 20018 San Sebastián, Spain.

Digital Object Identifier 10.1109/TBC.2020.3031724

I. Introduction

In THE last few years, the demand for video content across the Internet has constantly increased. Video streams from professional applications, such as Industrial Internet of Things (IIoT), medical equipment, connected and autonomous cars, and from domestic services, such as gaming, virtual reality, augmented reality, video over IP (VoIP) sports services, and over-the-top (OTT) platforms are flooding networks with real-time data intensive sessions. This evolution of Internet traffic makes evident the severity of the network's capacity to guarantee a certain quality of service (QoS) for the video applications. To prevent network flooding and to make video delivery more efficient, content delivery networks (CDNs) employ geographically distributed and cost-effective infrastructures as a service (IaaS) to enhance media availability and delivery performance across the Internet. This hierarchical system that caches and stores video streams fosters efficiency while geographical locations track human daytime life cycles, which have a close relation with local content demands.

Furthermore, the current video traffic crosses networks working on a best-effort basis where the delivery time of network packets is not guaranteed. Thus, it may cause stalls during playback on player devices, damaging the quality of experience (QoE). The popularity of video streaming services over the Internet pushed video industry-Moving Picture Experts Group (MPEG)-and standardization bodies to create new formats which enable adaptive streaming over the already existing Hypertext Transfer Protocol (HTTP) infrastructures. Thus, they allow the player devices to adapt the content representation to the specific device capabilities (resolution, codecs, etc.) and the changeable network connectivity.

Dynamic Adaptive Streaming over HTTP (MPEG-DASH) [1], which was designed to mitigate problems due to fluctuations on best-effort networks, is the solution adopted by the video industry [2]. In fact, MPEG-DASH enables pull-based streaming [3] and allows for scalable distribution as it has a CDN-ready design [4] that enables the exploitation of existing HTTP caching infrastructures without

modifications. To this end, the MPEG-DASH pipeline splits the video content into segments of fixed duration, usually between 2 and 10 seconds; then, it encodes them at different representation levels with a nominal resolution and bitrate. Thus, for each segment request, the player can switch from one representation to another depending on the assessed network status.

Nevertheless, the MPEG-DASH client-driven approach presents some drawbacks. First, each player is not aware of the existence of the others, leading to high network dynamics as the content download is not coordinated. Second, each player strives to achieve optimized individual quality, which may lead to unfairness when a congested connection path is shared [5]. Thus, it is challenging for a content provider (CP) to ensure a certain level of quality to end users, who are accessing large volumes of content through the same access point and competing for the available bandwidth independently. Here, some issues, such as initial buffering delay, temporal interruptions, unsteady video resolution, and bitrate changes, may damage the QoE [6].

Currently, CDNs are used to enhance media availability and delivery performance across the Internet. The proliferation of CDN vendors and business models allows the CP to use multiple CDN providers simultaneously [7], [8], [9]. However, extreme concurrency dynamics can affect CDN capacity, causing performance degradation and outages, while overestimated demand affects costs, thereby increasing the operational expenditure (OPEX) of the CP [10].

Upcoming 5G networks will need advanced and intelligent mechanisms to dynamically deliver each data flow according to the required service level agreement (SLA) and considering performance costs trade-offs. This concept is where the approach proposed by this article takes place, fusing network characteristics and media service options to match user satisfaction and business policies.

A. Contribution

This work proposes a novel solution called intelligent network flow (INFLOW) for CDN selection in a multi-CDN delivery environment. It exploits periodical MPEG-DASH media presentation description (MPD) updates to apply dynamic switching among the available CDNs at the video players in a standard compliant manner. The MPD with the appropriate CDN endpoint is served by the INFLOW Media Server, which works jointly with the INFLOW Forecast Service. The INFLOW Forecast Service provides network metrics predictions based on a Long Short-Term Memory (LSTM) network, a kind of Recurrent Neural Network (RNN), when fed with the historical values of network metrics. The integration of the Forecast Server into the delivery chain allows the Media Server to serve an MPD containing the *BaseURL* of the CDN, which fits target QoS and CP's business requirements. Thus, INFLOW allows for proactive and cost-effective video streaming delivery. The proposed solution comprises the following relevant contributions:

- Exploitation of network performance metrics and MPD information to apply common decisions to ongoing streaming sessions. Captured network metrics are employed to forecast CDN server capacity, then select a CDN only if it would guarantee the viability to serve the content at a minimum representation bitrate from the available ones in the MPD.
- Dynamic CDN server switching. We employ a dynamic approach switching from a CDN server to another depending on the performed predictions at any time. Thus, in contrast to current solutions, a streaming session is not served from a single CDN provider.
- Practical application of a forecast model. The literature proposing a forecast model for QoS network metrics is usually limited to theoretical analysis and simulations, and the predictions are not turned into video streaming actions. On the contrary, we exploit the predictions to switch the players among the available CDN servers, then proactively act on the delivery.
- Business constraints are considered for the CDN selection. We include metrics for both the OPEX and the QoS in the algorithm which selects the ideal CDN to be employed. Thus, this sophisticated approach favours the dynamic utilisation of a CDN marketplace to deal with cost-effective trade-offs. OPEX reduction, while keeping the QoS, is a major concern for practical deployments in real-world streaming services.

To achieve the above contributions, we develop a Forecast Service and a Media Server as complementary parts of the proposed INFLOW solution. Forecast Service executes an LSTM network performing real-time predictions of the QoS metrics. Media Server updates the MPD according to the network metrics predictions and CP's business rules and serves it to the clients. The solution was integrated and tested in a real setup employing a multi-modal testbed including both wired and wireless nodes. The wireless nodes were connected through a real Long-Term Evolution (LTE) RAN infrastructure of an operational Mobile Network stack including the radio base station (eNodeB) and the Evolved Packet Core (EPC). The traffic demand on video players was generated according to a probability distribution widely employed in the literature.

This article is structured as follows. First, Section II reviews related work in the field of video delivery based on CDN performance and network traffic generation and forecast. Then, Section III introduces the proposed INFLOW server, a novel media server equipped with a forecast service that tunes the delivery and applies a CDN selection mechanism based on QoS metrics and business rules, as the main focus of this article. Section IV describes the implemented setup using a real testbed, while Section V presents the results of the validation experiments. Finally, we assert our conclusions and future work in Section VI.

II. Related Work

A. CDN Resource Selection

A CDN is a network function widely employed to improve content delivery by means of cloud service provisioning cache

features. Fueled by the CDN vendor proliferation, media platforms exploit multi-CDN strategies to obtain more reliable content delivery that provides a steadier QoS and higher customer satisfaction. Nevertheless, the CDN selection criteria can be different for any CP.

A widely employed solution applies a static selection made by the media server when a new streaming session starts. This is used by Netflix [7] and Hulu [8], with big similarities [9]. They use three different CDN vendors mapping CDNs to the location of client device or to a subscriber. Moreover, they evidence that, the selected CDN is fixed during the streaming session even when the QoS degrades. Thus, providers are more prone to lower the representation bitrate instead of operating alternative CDNs. Hence, the authors conjecture that CDN selection is most likely based on business policies.

However, Netflix has changed its strategy over the years, and nowadays it uses its own CDN, which is called Open Connect [11]. Open Connect can be run inside the ISP infrastructure so that a better QoS can be achieved as the content is closer to the user. Netflix' solution is not that different from the open CDN architecture proposed by [12]. The authors propose collaborative participation of CPs and ISPs. On one hand, cost reductions are realized as the ISP acts as a CDN. On the other hand, the ISP provides better performance to the clients and reduces traffic as the content is already present in its network infrastructure.

The awareness of end-to-end QoS metrics measured by the client can make the difference when the employed CDN is dynamically chosen by the clients. The authors of [13] propose a client-side CDN selection. As a drawback, client-side strategies do not produce a coordinated decision as each client analyses the network performance of each CDN independently, introducing bias and communication overheads. Hence, a client-side CDN selection is not an optimal solution.

An intermediate solution consists of Domain Name System (DNS) resolution. Here, the DNS server can resolve a fixed hostname owned by the CP into different IP addresses of several CDNs. Depending on the DNS resolution, the client is directed to an appropriate CDN. The YouTube DNS-based solution is shown in [14]. YouTube goes further as it allows for the use of a hybrid DNS and application-level CDN selection. First, the DNS redirects the client to a server. Then, the server accepts or reject the client depending on the workload. If the client is refused, the DNS redirects the client to another server.

In [15], the effects of DNS resolution for CDN selection are further studied. The authors conclude that, depending on the DNS service provider, Akamai and Google CDN servers are chosen differently. Consequently, CDN performance highly depends on the load balancing rules of the DNS server. Here, a suboptimal CDN server selection leads to a higher round-trip delay time (RTT). To solve this problem, the authors propose a DNS-proxy running on the client. This proxy forwards the DNS requests to different DNS servers; then, it compares the responses to identify the best performing CDN server. However, these solutions are loosely coupled from media player requests slots applying balancing policies independently of media player timing. Instead, out approach is triggered by the requests with the most recent available information.

CDN Brokering [16] is the ability to redirect clients dynamically among two or more CDNs. CDN brokers collect and analyze the performance metrics of the available CDNs to select the one that performs best. Their work, in contrast to traditional multi-CDN strategies, is not limited to the selection of the initial CDN for each client. This solution also moves clients between CDNs when performance degradation is detected in real time. Thus, the CDN is dynamically and seamlessly changed. As an example, the European Broadcasting Union (EBU) proposed the EBU Flow Multi-CDN [17], which consists of a CDN switching service that selects the optimal CDN at any time. Similar approaches are provided by Citrix [18] and LightFlow [19]. Thus, these solutions are usually provided by intermediaries, federating infrastructures from different vendors. However, our approach keeps the control to the media service manager able to dynamically change the business policy or tune the cost function.

Edge computing systems promise a revolution on smart delivery of media traffic fueled by Multi-access Edge Computing (MEC) [20] architectures from 5G. Thus, new solutions for improving multi-CDN delivery involve investigating MEC services. In [21] a MEC proxy is proposed. The proxy can retrieve video streaming metrics of video players at the access point transparently and CDNs performance metrics from the wired link. Compared to a pure client-decision, a MEC proxy can evaluate the performance of each CDN just once and apply conclusions to other sessions (independently from the number of connected players). Moreover, it empowers the delivery through a local edge cache. This feature guarantees traffic reduction compared to server-side CDN selection as recurrent content can be downloaded and cached once for every client. In [22], a similar solution for a MEC-based cache is proposed. However, till the moment the edge systems realize, current CDN infrastructures makes the difference, and universal solutions that dynamically manage balancing are required.

In [23], a prototype of CDN and ISP collaboration is proposed. The ISP provides the CDN provider with services that allow the CDN provider to retrieve geographical user distribution and allocate server resources inside the ISP's network topology. The authors of [24] propose a similar solution without the binding to allocate resources in the ISP's infrastructure. In this case, a redirection center inside the ISP's network intercepts the client's requests and selects the appropriate CDN. This process is transparent to the client as the redirection center stores the content in a CDN surrogate and instructs an OpenFlow controller to migrate the traffic to a CDN surrogate.

Contrary to those approaches provisioning or balancing serving resources, other works focus on the selection of the appropriate bitrate to avoid congestion for a static number of servers [25] or network assets [26]. To this end, the MPD is parsed and heavier options are cropped. As implemented in these works, our approach employs a compliant MPD update mechanism. But, in our case, it is exploited to dynamically manage the CDN resources at any moment depending on network performance forecasts and CP business rules. The CP can tune the media server, thereby influencing CDN selection when the video player requests an MPD update.

TABLE I
FORECAST MODELS COMPARISON

Model	Approach	Number of variables	Internal parameters
ARIMA	statistical	univariate	a-priori (regression, integration and moving average parameters)
Exponential smoothing	statistical	univariate	a-priori (smoothing factor)
SETARMA	statistical	univariate	a-priori (regression, moving average and threshold delay parameters)
GARCH	statistical	univariate	a-priori (regression and lag length parameters)
Feed-forward NN	neural network	multivariate	trained (weight and bias)
RNN	neural network	multivariate	trained (input, output and forget factors)
LSTM	neural network	multivariate	trained (input, output and forget factors)

B. Time Series for Network Traffic Forecast

The goal of applying time series analysis to network traffic data is to forecast future conditions to take actions proactively when actuation performance or cost policies are satisfied. These techniques allow network management systems to prevent network under-performance and outages, thereby addressing network congestion preemptively.

The auto-regressive integrated moving average (ARIMA) is employed in [27] to predict the workload of cloud services. It employs historical records of observed requests to predict the volume of requests for the following time interval. The results reveal that the model can obtain the general trend, but it lacks the ability to accurately and timely track traffic peaks. The authors of [28] apply both ARIMA and exponential smoothing models to predict throughput in an LTE network. The two models are complementary, with ARIMA outperforming the exponential smoothing models on weekdays and the exponential smoothing models outperforming ARIMA on weekends.

The authors of [29] and [30] found limitations in ARIMA while modelling QoS attributes. QoS attributes, such as bandwidth or latency have nonlinear behaviors that do not fit the linear assumption of the ARIMA model. They overcame this by introducing hybrid linear and non-linear models. The linear model was represented by the ARIMA model. For the non-linear model, [29] used the self-exciting threshold autoregressive moving average (SETARMA) model, while [30] employed a generalized autoregressive conditional heteroscedastic (GARCH) model. In both cases, the proposed solution outperforms a standalone ARIMA model in forecasting the time between QoS violations.

In recent years, machine learning (ML)-based techniques for time series prediction have exhibited satisfactory performances. Specifically, neural networks (NNs) are gaining adoption in the generation of time series models. The authors of [31] propose a feed-forward NN for predicting the execution time of services while varying the number of requesters. In [32], a recurrent NN (RNN) is employed to forecast the end-to-end delay from RTT metrics.

In [33], an LSTM model, a particular type of RNN, was proposed. The authors employed a multivariate time series model where data input was probed from downlink control information (DCI) messages, such as resource blocks, transport block size, and scheduling information.

Table I shows the main differences between the employed techniques for time series forecasting. Clearly, NN-based approaches have the advantage to employ several variables as input and/or output of the models, while statistical ones are limited to one. Moreover, statistical approaches need a-priori evaluation of internal parameters, while NN-based ones are trained through a dataset of previous collected metrics. Here, the parameters for statistical approaches are intended for the whole model, while for NN-based ones, parameters must be trained for each internal cell.

From the available algorithms to analyze and predict time series, we employ LSTM network model as it satisfies two requirements. First, in terms of accuracy, statistical solutions (ARIMA and its derivatives) are slow when tracking quick fluctuations in time series as they tend to concentrate on the average value of the past observed values, as revealed in [34]. Second, in terms of multivariate time series, statistical solutions only can predict one variable. Thus, ARIMA would require separate models for forecasting both latency and bandwidth. On the contrary, LSTM is ready to process multivariate time series. The authors of [35] and [36] revealed that, the higher the number of input variables, the better the traffic predictions of LSTM when compared to ARIMA.

III. INFLOW SOLUTION

A. System Architecture

To achieve reliable and cost-effective video delivery, we propose the inclusion of our INFLOW solution in the video delivery chain. The overall scenario of the solution is depicted in Figure 1. The INFLOW solution is composed of two components:

- INFLOW Forecast Service: it receives the QoS performance metrics from the video players and processes them to predict the QoS values in the future.
- INFLOW Media Server: it exploits the results provided by the Forecast Service by combining them with the CP's business rules to select the appropriate CDN for each client.

Owing to the utilization of MPEG-DASH, INFLOW includes the following features:

- Scalability. New CDNs can be easily managed by adding them to the initial MPD.
- Real-time migration of video players to CDN providers. Supported by standard-compliant MPEG-DASH MPD

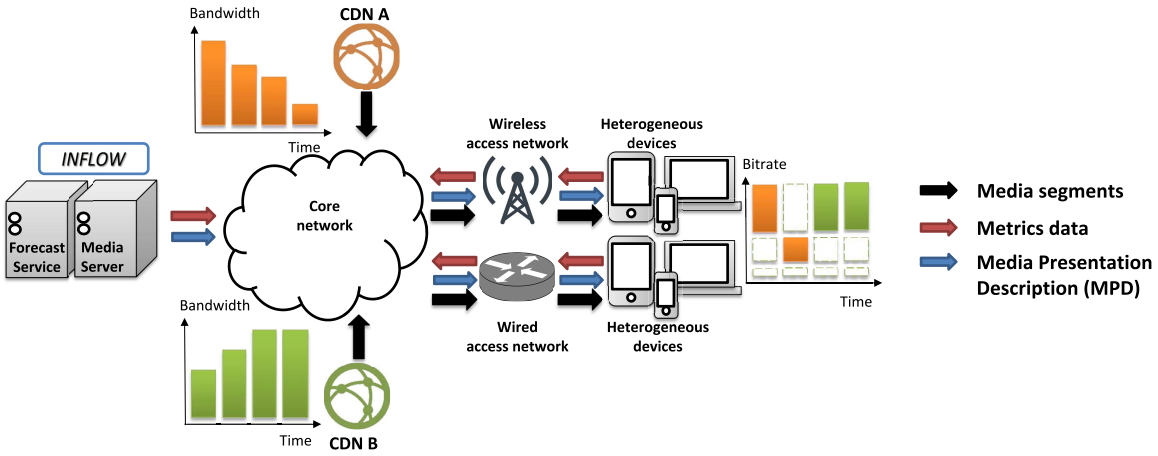

Fig. 1. General scenario of the proposed solution.

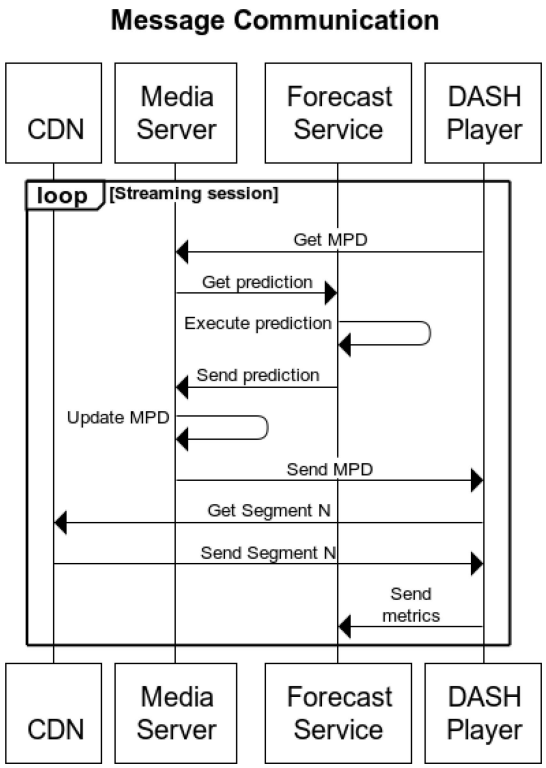

Fig. 2. Sequence diagram of the INFLOW solution for video delivery.

update mechanism manages the utilization of a CDN by the video player according to the gathered metrics and business policies.

- MPDs can be parsed and processed even when the content is encrypted with the MPEG-DASH Common Encryption Scheme (CENC) [37]. The CENC format encrypts the media segments indexed in the MPD, but the MPD is not encrypted.

The sequence diagram of the exchanged messages is depicted in Figure 2. Media segments are stored at different CDNs, while the MPD is served by the INFLOW Media Server. It is important that the media server uses a dynamic MPD as it forces the player to periodically update, overwriting the Minimum Update Period attribute from the MPEG-DASH standard [38]. On the other side, each video player downloads the initial MPD and starts requesting for segments from the initial CDN. A client-side adaptation mechanism constantly monitors the statistics of the downloaded segments to select a representation level among those available that fits with the experienced network performance. Thus, the video player aims to prevent stalls during playback. Typical monitored metrics are the network bandwidth and latency, which provide a direct measure of the QoS experienced by the client. Moreover, these measurements are sent to the INFLOW forecast service. Thus, video player should support a mechanism for sending feedback to the forecast service, such as Server and Network-assisted DASH (SAND) standard [39]. Finally, INFLOW forecast service stores the measurements and uses them to predict the future values.

The MPD served by the media server is fully conditioned by the prediction of the forecast service. Every time a player requests an updated version of the MPD, the media server retrieves a prediction from the forecast service and decides to serve the current MPD or to change the CDN included. Therefore, the forecast service does not apply any QoS or business rules–it simply processes the information provided by the players. The QoS and business-based decisions are made by the media server, and this decision process is executed in real-time as the predictions are rendered out of date after the predicted interval, leading to a new decision. Therefore, the shorter is the segment duration, the more immediate is the forecast validity and the prompter is the MPD update.

The QoS forecasts serve two roles. First, the INFLOW Media Server can select an appropriate CDN to shield from CDN service degradation and outages based on the most recent detected performance. To this end, the server receives alternative CDNs from the initial MPD and replaces the *BaseURL* tag in the MPD with another CDN endpoint to migrate a client. Second, the media server can count the video sessions served by each CDN. On top of this information, the media server can apply cost-effective policies, allocating extra CDN resources to enforce QoS or retiring CDN assets to reduce the number of employed CDN servers. Thus, the media

Algorithm 1 INFLOW Forecast Service

function PREDICTMETRICS($\overline{bw^k_{t-1}}$, $\overline{l^k_{t-1}}$, CDN^k)
▷ for each CDN infrastructure
Input: $\overline{bw^k_{t-1}}$ ▷ bandwidth mean for the most recent period @CDN^k
Input: $\overline{l^k_{t-1}}$ ▷ latency mean for the most recent period @CDN^k
Output: $\widehat{bw^k_t}$ ▷ bandwidth prediction
Output: $\widehat{l^k_t}$ ▷ latency prediction
$\{bw^k\} = \{\overline{bw^k_{t-1}},...,\overline{bw^k_{t-N}}\}$ ▷ N bandwidth samples
$\{l^k\} = \{\overline{l^k_{t-1}},...,\overline{l^k_{t-N}}\}$ ▷ N latency samples
$\widehat{bw^k_t},\widehat{l^k_t} \leftarrow$ LSTM($\{bw^k\},\{l^k\}$) ▷ update forecast model CDN^k

service can manipulate the OPEX ranges to meet the business model.

In the following section, we describe separately the two components of the INFLOW solution.

B. INFLOW Forecast Service

The INFLOW forecast service is in charge of collecting network metrics probed and sent by the video players and processing them to predict the values in future slots. The most recent metrics are processed while older ones are discarded using a sliding window mechanism. The decision program of the forecast service is shown in Algorithm 1.

The input are the last N historical values of network bandwidth and latency measured and reported by the players for a specific CDN (CDN^k). The samples comprised in the most recent period are captured during last segment download. The video player requests segments in a regular pace to fill its buffer, according to the media duration of the segment. In total, the algorithm processes two variables taken in N time instants. Every new sample should be taken at a fixed temporal distance from the previous one. Nevertheless, this assumption of equal distance among the samples is not guaranteed as players usually run asynchronously, and then the reports are sent in a random time inside a segment slot. To overcome this problem, the INFLOW forecast server employs a mean value of samples within a second as the input of the algorithm ($\{\overline{bw^k_{t-1}}, \ldots, \overline{bw^k_{t-N}}\}$ and $\{\overline{l^k_{t-1}}, \ldots, \overline{l^k_{t-N}}\}$). The input is processed through the LSTM network to predict the values in the next second ($\widehat{bw^k_t}$ and $\widehat{l^k_t}$). The predicted values are the output of the algorithm.

It is important to underline that the benefits of an LSTM network over statistical approaches are two-fold. First, the LSTM network performs better when time series includes quick fluctuations [34]. Second, it is valid for multivariate time series, such as bandwidth and latency, where statistical methods fail to simultaneously process several components [35], [36].

C. INFLOW Media Server

The media server must serve the MPD of the video players to provide awareness on available representations, content formats and metadata, and CDN endpoints. In our case, as we were interested in CDN localization, the served MPD could include one or more *BaseURL* tags containing the URLs of the CDN servers. In cases with only one *BaseURL*, the client is forced to use it.

Algorithm 2 INFLOW Media Server

function UPDATEMPD(*urlMPD*, *SLA*)
▷ for each MPD request
Input: *urlMPD* ▷ requested MPD
Input: *SLA* ▷ applicable SLA
Output: *MPD* ▷ updated MPD
MPD ← initial(*urlMPD*) ▷ requested MPD file
bw_{min} ← targetQoS(*SLA*) ▷ minimum bandwidth per player
d_s ← *MPD* ▷ segment duration
$\{CDN_{list}\}$ ← *MPD* ▷ set of alternative CDNs
for all $CDN^k \in \{CDN_{list}\}$ **do** ▷ for each CDN
$\overline{bw^k_{t-1}} \leftarrow$ mean($\{bw^k_{[t-1,t)}\}$) ▷ average for most recent period @ CDN^k
$\overline{l^k_{t-1}} \leftarrow$ mean($\{l^k_{[t-1,t)}\}$) ▷ average for most recent period @ CDN^k
$\widehat{bw^k_t},\widehat{l^k_t} \leftarrow$ predictMetrics($\overline{bw^k_{t-1}},\overline{l^k_{t-1}},CDN^k$)
$n^k \leftarrow$ sessions(CDN^k) ▷ total CDN^k sessions
$\widehat{n^k} \leftarrow$ policy($\widehat{bw^k_t},\widehat{l^k_t},n^k,bw_{min},d_s,CDN^k$)
▷ CDN^k capacity
if ($\widehat{n^k} > n^k$) **then** ▷ CDN^k admits more sessions
BaseURL ← URL(CDN^k)
▷ write CDN^k URL
MPD ← update(*MPD*,*BaseURL*)
▷ update MPD

The INFLOW media server stores an initial MPD containing different *BaseURL* tags and modifies it while excluding CDN alternatives to force a CDN to perform according to the algorithm outcomes, which exploits the predictions provided by the INFLOW forecast service. The decision program of the media server is shown in Algorithm 2.

The algorithm takes an initial configuration of minimum bandwidth to be provided to the clients (bw_{min}) according to the SLA, and the initial MPD. From the MPD, it retrieves the segment duration (d_s) and a list of the CDNs ($\{CDN_{list}\}$). When an MPD request reaches the media server, it selects an appropriate CDN (CDN^k) from the CDN list. This list ($\{CDN_{list}\}$) is ordered in ascending order according to expenses. Thus, the media server first employs the affordable providers, migrating users to cheaper services when possible. The media server retrieves the prediction for each CDN from the forecast service ($\widehat{bw^k_t}$ and $\widehat{l^k_t}$) and stops if the expected capacity ($\widehat{n^k}$) is higher than the current ones (n^k). In other words, it selects the most affordable CDN that has the capacity to serve more players. The number of expected players is evaluated through the predictions and the initial configuration by means of

Equation (1).

$$\widehat{n^k} = \frac{\left(d_s - \widehat{l_t^k}\right) * \widehat{bw_t^k} * n^k}{d_s * bw_{min}} \quad (1)$$

To be timely delivered, the theoretical maximum download time of a segment should be lower than the segment duration. Furthermore, a padding time must be considered to take into account the delay introduced by the network during the transmission. Consequently, the predicted latency ($\widehat{l_t^k}$) is used as a penalization factor to estimate the effective download time (d_s-$\widehat{l_t^k}$). Then, this value is multiplied by the predicted average bandwidth per video player ($\widehat{bw_t^k}$) to assess the average volume of data that each player can download. The total data capacity is obtained by multiplying the number of sessions in the CDN (n^k) by the average volume of data that each player can download. Finally, the overall traffic demand is divided by the amount of data that a player should download during a segment duration (d_s) according to the SLA using the minimum bandwidth provided (bw_{min}). The final value is the CDN capacity according to an SLA, which is indicative of the number of video streaming sessions that the CDN can serve ($\widehat{n^k}$).

Once a CDN is assigned to a session, the media server selects the *BaseURL* corresponding to the CDN and generates a new MPD by modifying the initial one. The order of the CDNs in the list is important as they are ranked depending on the cost. Thus, the media server chooses the first CDN that fits with the necessary resources; then, the most affordable ones are quickly booked to reduce CP's OPEX.

IV. Testbed Setup

To demonstrate the cost-effective advantages of the INFLOW approach in terms of QoS enforcement and CP's OPEX reduction, we deployed a heterogeneous and distributed setup employing both FED4FIRE+ facilities [40] and our facilities at Vicomtech (San Sebastián, Spain). Fed4FIRE+ is a Horizon 2020 project that provides open and accessible testbeds to support research and innovation initiatives in Europe. Among the available facilities, we employed NITOS's network infrastructure [41] at University of Thessaly's campus (Volos, Greece). NITOS provides heterogeneous testbeds to execute experiments on real wired and wireless networks.

We use D-DASH dataset and infrastructure [42], with Dynamic Adaptive Streaming over HTTP (DASH) standard content mirrored over different sites at different locations to perform CDN-based scientific evaluations. The dataset includes the Red Bull Playstreet video sequence, which is owned by Red Bull Media House and licensed for scientific purposes. This sequence is encoded for 17 video representations through advanced video coding (H.264/AVC) and 4 dual channel audio representations through advanced audio coding (AAC). Both audio and video are segmented with different segment lengths of 2, 4, 6, 10, and 15 seconds, and multiplexed in ISO MPEG4 files (ISO/IEC 14496-12 - MPEG-4 Part 12). For our experiments, we employed 2 seconds segments to focus on live video content where dense client cells and congestion of CDNs were likely. We did not modify the video representations; instead, we used the available representations in the dataset. The representations range from a resolution of 320 x 240 and 30 fps at 100 kbps to a resolution of 1920 x 1080 and 30 fps at 6000 kbps. As the client-side bitrate adaptation mechanism works on a best-effort basis and do not take care of the presence of other connected players, each player struggles to achieve the highest representation bitrate.

The final experimental setup comprises the following:

- 4 UE nodes: client nodes located at NITOS and running 100 DASH video players based on GStreamer multimedia framework [43]. They feature both Ethernet and LTE interfaces and are placed in the isolated environment of the NITOS indoor testbed where they form a grid topology.
- 1 eNodeB: USRP provided node performing eNodeB stack located at NITOS. It forwards the packets from the clients to the Access and Core Network.
- 1 EPC node: wired node close to the eNodeB that executes the EPC stack.
- 1 INFLOW Media Server: node at Vicomtech based on a virtual machine with 2 GB RAM and single-core CPU. It is provided with a public IP address to serve the MPD to the video players. It runs a Node.js [44] server application which applies QoS and CP's business rules when sending the MPD to the client.
- 1 INFLOW Forecast Service: node at Vicomtech based on a physical machine having 12 GB RAM and quad-core Intel i5 6500 CPU. To perform predictions, it features NVIDIA GTX 1050 TI executing the LSTM model based on TensorFlow [45].
- 3 servers: they belong to the D-DASH dataset [42] providing alternative CDNs storing the media segments to perform CDN-based scientific evaluations. They are located at different sites with different nominal performances in terms of bandwidth and latency.

To distribute the video streaming sessions between the wired and LTE network interfaces, we considered the last Cisco report concluding that mobile traffic covers 9% of the total IP video traffic [46]. Hence, the experiment setup includes nine video players connected through the LTE interface and 91 players employing Ethernet interface. The use of different access networks is helpful for demonstrating its applicability in representative and multi-modal scenarios. Moreover, we modeled player inter-arrival rate and session duration according to [47], which provides an extensive analysis on user behavior while accessing streaming services. Thus, the inter-arrival time distribution is a modified version of the Poisson distribution, while the session duration follows the declared sections of 5 (37.44%), 10 (52.55%), or 25 min (75.25%).

During the experiment, a preliminary step was performed to generate a QoS performance metric dataset for training our LSTM model at the Forecast Service. The setup for dataset creation is depicted in Figure 3(a). Here, the Media Server serves a static MPD. It does not allow for player migration among the different CDNs so that full characterization of a specific CDN can be achieved, resulting in a time series for a CDN. Then, during streaming sessions, network bandwidth and latency measurements provided by the video players are

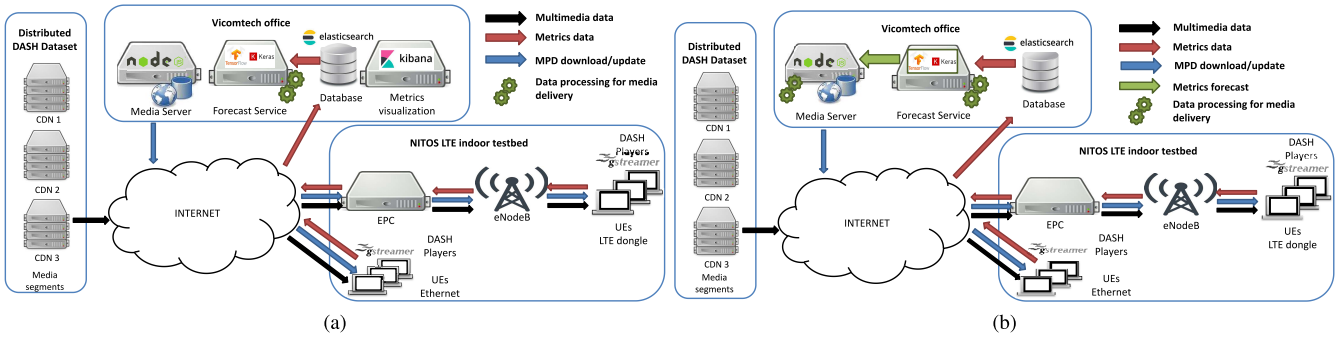

Fig. 3. Testbed setup: configurations for dataset creation (**a**) and for INFLOW enabled delivery (**b**).

stored in an Elasticsearch [48] database and employed to train the forecast service predictor. Optionally, Kibana [49] dashboards are available to visualize the collected metrics and guide LSTM training and tuning.

After the training phase is completed, a new setup for testing the proposed INFLOW solution is applied (3(b)). Now, the collected metrics sent with a SAND-alike mechanism are consumed by the forecast service to execute bandwidth and latency predictions for the next period. The predictions are employed by the media server to apply its decision rules for CDN selection. In this case, the media server serves an MPD dynamically updated to force video players to periodically request it. The update period is equal to the segment duration (2 seconds) and it is set through the *minimumUpdatePeriod* tag inside the MPD.

In this setup, we aimed to compare INFLOW with other CDN selection strategies. Then, we compared the results for the following common CDN selection strategies:

- *Single CDN (SC):* this experiment does not involve multiple CDNs. It uses just the most affordable CDN for all the clients.
- *Equal selection (ES):* this experiment consists of balancing the occupancy rate of each CDN assigned when the session starts. Therefore, every CDN has the same number of connected clients, and the video players do not migrate between CDNs.
- *Progressive selection (PS):* this experiment consists of progressive allocation of new CDNs when the used one(s) gets exhausted, i.e., when the theoretical maximum number of connected clients is reached and the bandwidth from the SLA is consumed. The maximum number of clients is set to 33 (100 players/3 CDNs). The clients do not migrate between CDNs.
- *INFLOW selection (INFLOW):* this experiment exploits the capabilities of the proposed INFLOW solution to dynamically migrate the clients depending on the predictions and the applicable cost ranges. It aims to minimize the use of CDN providers at any moment.

It is important to note that INFLOW needs to be set with the SLA for the clients to avoid any violation on QoS. Setting a bandwidth threshold lower than the minimum representation bitrate (100 kbps) is useless as the players should always experience at least the minimum representation bitrate to play the content. In the same way, a value higher than the maximum representation bitrate is not valid. Then, we decided to set the minimum bandwidth to 4 Mbps; this was enough to play a smooth 1080p video, which corresponds to two-thirds of the maximum available representation bitrate (6 Mbps).

V. Validation and Results

A. Predictor Validation

The generation of the LSTM model consisted of three steps: training, validation, and testing. Both the training and validation steps employed a training dataset, where 80% of the samples were used for training and the remaining 20% were used for the validation step. The training dataset consisted of a multivariate time series, and bandwidth and latency measurements were taken for three hours. The collection was performed in three different sessions lasting one hour each. Each session was executed on a different day and employed a different CDN to download the content. The testing process employed a testing dataset. The testing dataset consisted of the training dataset with an extra hour of data collected on a different day that was independent of the training dataset.

To guarantee that the LSTM model used equal spaced input measurements, the simple moving average (SMA) was applied to both datasets so that an average bandwidth and latency value could be computed each second. This resulted in 10800 samples for the training dataset and 3600 samples for the testing dataset. A total of 8640 samples of the training dataset (80%) constituted the training set, while the remaining 2160 samples (20%) were used as the validation set.

The training set was employed in the first phase to generate the LSTM model. The autocorrelation plot, depicted in Figure 4, shows a clear correlation of the tuple (bandwidth, latency) in the time series. Here, the autocorrelation is lower for samples that are more distant. Consequently, samples which are closer to the one we want to predict are the most valuable. The LSTM model provides next values based on the last N bandwidth and latency measurements. N has been empirically set to 7. A shorter window had a big impact on LSTM accuracy, while a longer one did not result in a significant increase in LSTM forecast fidelity. The accuracy results when $N = 6$ dropped by 0.2% for the bandwidth and by 2.7% for the latency. The accuracy increased when $N = 8$ was under 1% for both time series.

A comparison of the values measured and predicted during the validation is shown in Figure 5. The graphs show that the

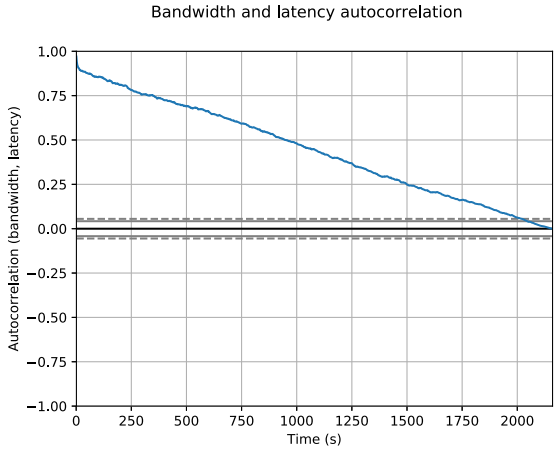

Fig. 4. Bandwidth and latency autocorrelation.

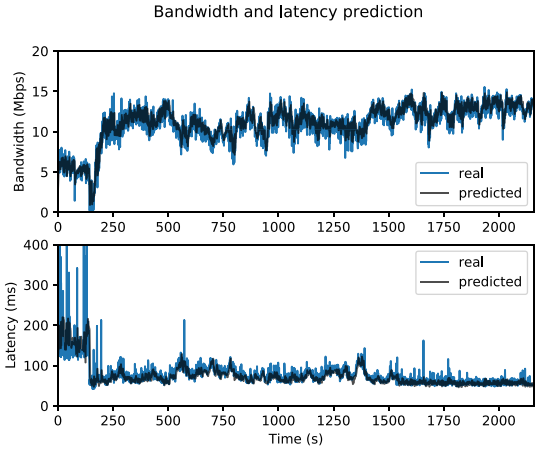

Fig. 5. Bandwidth and latency prediction: validation of the LSTM model.

predictor can follow the trend of the time series, but it cannot predict sudden and drastic changes (high or low outliers). We calculated the mean absolute error (MAE) and the root mean square error (RMSE) for both bandwidth and latency. The MAE values were 0.76 Mbps and 11 ms for bandwidth and latency, respectively, while the RMSE values were 0.99 Mbps and 27 ms, respectively.

Once the model was validated, we generated the final model by training it with all the training dataset (10800 samples). Then, the final model was employed to predict values of the testing dataset. We limited the test to 2160 samples to foster a fair comparison with the validation results with a similar number of samples. For this subset, we compared the obtained values of MAE and RMSE with the ones coming from the previous validation. Figure 6 shows the results of the testing process. The bandwidth MAE and RMSE were equal to 0.94 Mbps and 0.51 Mbps, respectively, which are definitely close (and even better) to the values obtained during the validation. On the contrary, the latency MAE and RMSE were 31 ms and 97 ms, respectively, making evident that the latency is harder to accurately predict. From the Figure 6, it is clear that latency produces higher outliers than the bandwidth, which are difficult to predict.

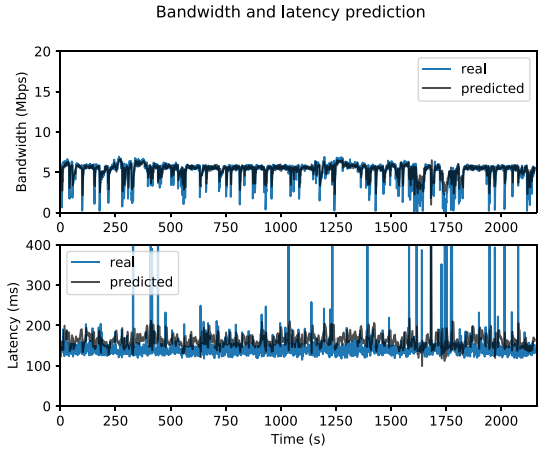

Fig. 6. Bandwidth and latency prediction: testing of the trained LSTM model.

B. QoS Performance Comparison

INFLOW aims to manage the QoS performance and business cost trade-off. To this end, we identified different performance metrics for both the parameters to evaluate and balance them. We carried out the QoS evaluation by collecting the representation bitrate selected by the adaptation algorithm of the video players. Moreover, we compared the representation bitrate with the measured network bandwidth and latency to evaluate the efficiency of the utilization of the CDN resources as the efficiency increases as the overall throughput of a CDN approaches the available CDN bandwidth.

We tested our solution by comparing it with other CDN selection strategies. The final set of experiments utilized the *single CDN* (*SC*), *equal selection* (*ES*), *progressive selection* (*PS*), and *INFLOW selection* (*INFLOW*) strategies. As mentioned in the previous section, the minimum bandwidth for *INFLOW selection* algorithm was set to 4 Mbps, and the max amount of running players for each experiment was 100.

Table II shows the network latency for the video players. The results for each CDN while employing *SC*, *ES*, and *PS* strategies are close to each other. Each CDN presents similar latency independently of the strategy. On the contrary, *INFLOW* strategy presents a higher latency of up to +124% (CDN3) as switching the connection from a CDN to another inevitably implies the addition of delay. Furthermore, if the experienced latency is still in the order of hundreds of milliseconds, then it does not affect the video players, which have a playback buffer equal to one segment duration (2 seconds).

Table III shows the available network bandwidth for the video players while video content is being downloaded from the CDNs. Table IV presents the selected bitrate from the client-side algorithm.

SC strategy provides only information for CDN1 as the other two are never used. This strategy provides the worst results when compared to the others because the players are experiencing a highly congested CDN communication. The average measured bandwidth is 2.54 Mbps, and average representation bitrate is 1.67 Mbps.

As expected, *SC* results improve when the multi-CDN strategy comes into place, therefore allowing for load balancing.

TABLE II
AVERAGE VALUE AND STANDARD DEVIATION OF THE MEASURED LATENCY BY THE PLAYERS

Strategy	CDN1		CDN2		CDN3	
	l_{avg}(ms)	l_{dev}(ms)	l_{avg}(ms)	l_{dev}(ms)	l_{avg}(ms)	l_{dev}(ms)
Single CDN	89	39	-	-	-	-
Equal selection	89	42	132	35	42	25
Progressive selection	85	35	127	23	51	157
INFLOW selection	114	68	125	54	94	75

TABLE III
AVERAGE VALUE AND STANDARD DEVIATION OF THE MEASURED BANDWIDTH BY THE PLAYERS

Strategy	CDN1		CDN2		CDN3	
	bw_{avg}(Mbps)	bw_{dev}(Mbps)	bw_{avg}(Mbps)	bw_{dev}(Mbps)	bw_{avg}(Mbps)	bw_{dev}(Mbps)
Single CDN	2.54	0.44	-	-	-	-
Equal selection	2.66	0.41	15.50	7.16	11.42	2.76
Progressive selection	2.90	0.34	21.84	4.79	10.65	2.17
INFLOW selection	3.52	2.43	5.23	3.85	5.88	2.82

TABLE IV
AVERAGE VALUE AND STANDARD DEVIATION OF THE SELECTED BITRATE BY THE PLAYERS

Strategy	CDN1		CDN2		CDN3	
	R_{avg}(Mbps)	R_{dev}(Mbps)	R_{avg}(Mbps)	R_{dev}(Mbps)	R_{avg}(Mbps)	R_{dev}(Mbps)
Single CDN	1.67	0.62	-	-	-	-
Equal selection	1.78	0.67	4.46	2.10	4.77	1.71
Progressive selection	1.96	0.61	5.10	1.78	4.48	1.98
INFLOW selection	1.96	1.23	2.44	1.57	2.82	1.43

ES and *PS* strategies limit to 33 (100 players / 3 CDNs) the number of players connected to CDN1. Both strategies produce similar results for the different CDNs. For CDN1, the average measured bandwidth of *ES* and *PS*, when compared to *single CDN* strategy, improves +4.7% and +14.1%, respectively. These results mean a higher bitrate selection, +6.6% and +17.4%, respectively. Moreover, the selected bitrates are considerably higher for CDN2 and CDN3 as they provide a higher performance. These CDNs provide more network resources serving higher bandwidths for video players. The measured results range between 15.50 Mbps (*ES*) and 21.84 (*PS*) for CDN2 and 10.65Mbps (*PS*) and 11.42 (*ES*) for CDN3. Thus, distributing video players across the available CDNs by just considering the number of players per CDN (33 players) is not fair as video players connected to CDN2 and CDN3 can select higher representation bitrates. Video players select a representation bitrate up to +160% higher if connected to CDN2 and +168% higher if connected to CDN3.

INLFOW uses a different approach. Here, the maximum number of players for each CDN is constantly updated through Equation (1). Then, video players are dynamically switched at any time. CDN1 still underperforms compared to CDN2 and CDN3, but the fairness is lower. The residual performance bias is due to the fact that CDN1 is still the preferred CDN, i.e., the other two are not used until CDN1 is congested. Accordingly, CDN3 is not used until CDN2 is congested too. Here, the higher average measured bandwidth is 5.88 Mbps at CDN3, which is +67% higher than the result at CDN1 (3.52 Mbps). For *ES* and *PS*, the variations are +483% (CDN2 compared to CDN1) and 653% (CDN2 compared to CDN1), respectively. In terms of the selected representation bitrate, *INFLOW* keeps the results obtained by the other multi-CDN strategies at CDN1 but underperforms at CDN2 and CDN3. This is because INFLOW aims to reduce the number of employed CDNs and the OPEX at any time, while guaranteeing at least 4 Mbps for the measured network bandwidth. Thus, video sessions to CDN2 or CDN3 can be retired, therefore saving the CDN OPEX. The other multi-CDN strategies do not reduce CDN usage, i.e., when the player is connected to a CDN, the connection is maintained until the session expires. In terms of fairness, *INFLOW* outperforms the other multi-CDN strategies as the average representation bitrates of each CDN are close to each other. Video players connected to CDN2 present the best representation bitrate (2.44 Mbps), which is +24% higher than those of video players connected to CDN1. Moreover, standard deviation for measured bandwidth and representation bitrate achieved with INFLOW also demonstrates that it is the fairest solution since the players experience almost the same variation independently of the CDN. Standard deviation for CDN2 is +58% higher than CDN1 if we consider measured bandwidth and +27% if we consider representation bitrate.

Concerning communications overheads to proactively enforce QoS, the traffic overhead is 893 MB from the total traffic (53592 MB). Thus, overhead causes an increase of +1.6% in the transmitted data. INFLOW exploits MPD update mechanism with an update period is equal to the segment duration. In our case this means 9040 Bytes requested every 2 seconds by each player. In the other strategies, MPD update mechanism is not used, then there is not an additional overhead.

The predictions performed by the INFLOW Forecast service during the MPD requests causes also higher MPD delivery delay compared to the other strategies. As a result, INFLOW strategy adds 53 ms of delay while delivering the MPD. Nevertheless, this delay does not affect the playback since the

player employs the previous MPD until a new one is received and parsed. Thus, if it is necessary to perform a segment request during an MPD update, it is performed in any case, as the two operations are executed in different threads and do not interfere with each other. In terms of resource utilization, the node running the INFLOW Forecast service shown 3.2 GB RAM, 33% CPU and 27% GPU peak utilization rates. Thus, its hardware configuration could absorb a larger number of users.

C. Business Cost Comparison

Regarding business cost, OPEX consists of ongoing expenses that a business incurs inherent to the operation of the assets. In our case, we were interested in OPEX, as we wanted to evaluate the cost of the CDN resources due to their ongoing utilization, i.e., it depends on the utilization of the CDN resources at any time [50].

There is no common formula for evaluating the OPEX, as in many cases, the provider does not publish publicly its pricing plans, but it offers personalized plans to each customer. Nevertheless, [51], [52] and [53] reveal that the OPEX for CDN resources depends on a set of factors, such as the employed network, storage, and time resources. Then, we express monthly OPEX through Equation (2).

$$OPEX_{month} = \sum_{i=1}^{K} \alpha_{loc_i} * Tr_i + \beta_{loc_i} * K_{req_i} + \gamma_{loc_i} * T_i + \delta_{loc_i} * St_i + \epsilon_{loc_i} \quad (2)$$

In the equation, Tr_i is the traffic volume in a month, K_{req_i} is the number of HTTP requests producing such demand, T_i is the utilization time for a CDN that has active sessions from video players of a service, and St_i is the employed storage at the CDN. Therefore, α_{loc_i}, β_{loc_i}, γ_{loc_i}, δ_{loc_i}, and ϵ_{loc_i} are multiplicative coefficients established by a particular CDN provider and that depend on the location of the resources (cost of the servers depends on the country where they are located). The addition indicates that we are in multi-CDN environment. Then, we need to sum over the K available CDNs.

The values for the coefficients are closely related to the business model and the pricing plan of each CDN provider. Accordingly, the monthly OPEX is tailored to the employed CDNs. In any case, we evaluate the variables independent of the CDN vendor, which depend on the resources we are employing during the tests (Tr_i, K_{req_i}, T_i and St_i) and directly impact the OPEX.

To simplify the evaluation of OPEX, we have made some assumptions. First, St_i is fixed for each experiment as the amount of employed storage depends on the content size, and it is permanently stored, even if it is never requested. Second, K_{req_i} is almost constant as, in any case, the experiments run 100 players, which request a media segment and an MPD every 2 seconds (segment duration). Third, Tr_i is directly proportional to the selected representation bitrate. It can be roughly calculated by multiplying the mean bitrate of the sessions from video players and the duration of the experiment. As the selected bitrate is already being captured for the QoS evaluation, we can assess the traffic volume. Finally, T_i is the variable that really changes across every experiment depending on the cost-effective strategy, leading to different utilization rates for each available CDN. Then, we employ T_i as the main metric for comparing the OPEX achieved by the different strategies.

TABLE V
UTILIZATION TIME OF THE CDNS

Strategy	CDN1	CDN2	CDN3
	T_i(minutes)	T_i(minutes)	T_i(minutes)
Single CDN	60	-	-
Equal selection	59	59	58
Progressive selection	58	58	57
INFLOW selection	52	48	31

Table V shows the usage time of each CDN while applying the different strategies. *ES* strategy is the most expensive solution as all the CDN are utilized almost all the time. In this case, the overall usage time is close to 3 h (1 h per each CDN). The actual result is 176 min. On the contrary, *SC* results in lower business costs as CDN2 and CDN3 are never employed. In this case, the usage time is just 60 min. *PS* is quite like *ES*. Figure 7 shows the number of players connected to each CDN for one hour and it is clear that *ES* and *PS* differ only in the first minutes. In *ES*, the three curves increase almost together, while in *PS*, the curves separately increase because CDN2 is not employed until CDN1 reaches 33 players and CDN3 is only employed after both CDN1 and CDN2 reach 33 players. The overall usage time is 173 min, which corresponds to a reduction of -2% compared to *ES*. From Table V, the usage time of each CDN while employing PS strategy is similar to ES one. Finally, *INFLOW* graph presents a completely different behavior. Here, the number of players connected at each CDN is much more variable owing to the switching mechanism. The number of players of each CDN ranges between 0 (the CDN is not being used) to 100 (CDN serving all the players). The number of players for the other multi-CDN strategies is always around 33 players. Nevertheless, INFLOW can retire the sessions from a CDN, which is not necessary after migrating the clients. This results in 131 min of overall usage time; then the reductions for *ES* and *PS* are -26% and -24%, respectively. Compared to *SC* strategy, *INFLOW* employs $+118\%$ more CDN usage time, while the value increases to $+193\%$ and $+183\%$ for *ES* and *PS*, respectively. If we focus on the usage time of each CDN, Table V clearly shows that INFLOW reduces the usage of CDN2 and CDN3.

In summary, the proposed INFLOW solution improves the CDN resource management by dynamically selecting the CDN for each video player at any time. It allows for business cost saving by decreasing the usage time of the available CDNs, while maintaining a minimum bandwidth level. Moreover, the resources are more efficiently exploited because the players are distributed depending on the real capabilities of each CDN, such as the experienced network resources. Consequently, the selected bitrate is fairer among the players.

VI. CONCLUSION AND FUTURE WORK

The trend for the following years is an increasing consumption of media content, where the content is mostly delivered

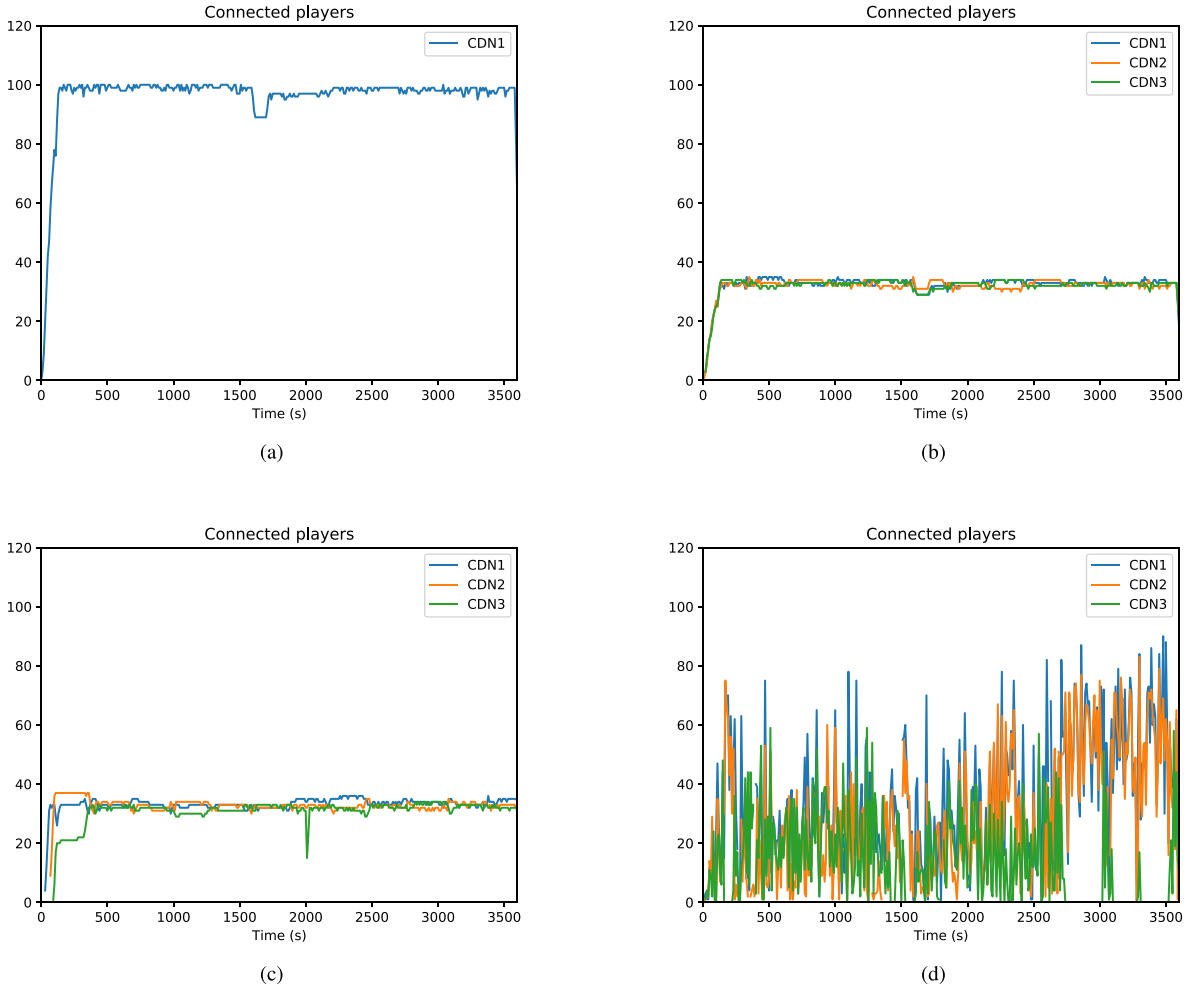

Fig. 7. Distribution of the players among the available CDNs: single CDN (**a**), equal selection (**b**), progressive selection, (**c**) and INFLOW selection (**d**) strategies.

through CDN infrastructure. Here, the CP strives to guarantee the necessary QoS for its media service while reducing the business costs associated with the CDN.

Toward this goal, we introduce a novel solution called INFLOW for CDN selection in a multi-CDN delivery environment. INFLOW enables the media server with a forecast service so that metrics collected by the player are processed as MPEG-DASH streams are served. The forecast service executes time series analysis through an LSTM model for prediction of the future values of network bandwidth and latency. The predictions are exploited by the media server to act while the player requests an MPD update. The media server can decide to keep the same MPD or change it to switch the player to another available CDN from which content can be downloaded.

The proposed solution has been implemented and validated in a distributed and heterogeneous testbed employing real network nodes. The evaluation includes a comparison with other CDN selection strategies in terms of QoS and business cost. The results highlight the advantages of INFLOW for reducing the overall usage time of the available CDNs, while guaranteeing a minimum level of network bandwidth to every player.

Future work includes the exploitation of new metrics to improve the predictions made by the forecast service. Moreover, the collected metrics can be further exploited to obtain an estimation of a user's QoE, and actions that also take into consideration the user's expectations can be taken.

REFERENCES

[1] I. Sodagar, "The MPEG-DASH standard for multimedia streaming over the Internet," *IEEE MultiMedia*, vol. 18, no. 4, pp. 62–67, Apr. 2011, doi: 10.1109/MMUL.2011.71.

[2] K. Park, N. Kim, and B. D. Lee, "Performance evaluation of the emerging media-transport technologies for the next-generation digital broadcasting systems," *IEEE Access*, vol. 5, pp. 17597–17606, 2017.

[3] A. Begen, T. Akgul, and M. Baugher, "Watching video over the Web: Part 1: Streaming protocols," *IEEE Internet Comput.*, vol. 15, no. 2, pp. 54–63, Mar./Apr. 2011. doi: 10.1109/MIC.2010.155.

[4] P Maillé and G. Schwartz, "Content providers volunteering to pay network providers: Better than neutrality," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2016, pp. 484–489, doi: 10.1109/INFCOMW.2016.7562125.

[5] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis, "What happens when HTTP adaptive streaming players compete for bandwidth?" in *Proc. 22nd Int. Workshop Netw. Oper. Syst. Support Digit. Audio Video (NOSSDAV)*, 2012, pp. 9–14, doi: 10.1145/2229087.2229092.

[6] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, "A survey on quality of experience of HTTP adaptive streaming," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 469–492, 1st Quart., 2015, doi: 10.1109/COMST.2014.2360940.

[7] V. K. Adhikari *et al.*, "Unreeling netflix: Understanding and improving multi-cdn movie delivery," in *Proc. IEEE INFOCOM*, 2012, pp. 1620–1628, doi: 10.1109/INFCOM.2012.6195531.

[8] V. K. Adhikari, Y. Guo, F. Hao, V. Hilt, and Z. L. Zhang, "A tale of three CDNs: An active measurement study of Hulu and its CDNs," in *Proc. IEEE INFOCOM Workshops*, 2012, pp. 7–12.

[9] V. K. Adhikari *et al.*, "Measurement study of NetFlix, Hulu, and a tale of three CDNs," *IEEE/ACM Trans. Netw.*, vol. 23, no. 6, pp. 1984–1997, Dec. 2015, doi: 10.1109/TNET.2014.2354262.

[10] S. D. Silva, J. Bruneau-Queyreix, M. Lacaud, D. Negru, and L. Reveillere, "MUSLIN: A QoE-aware CDN resources provisioning and advertising system for cost-efficient multisource live streaming," *Int. J. Netw. Manag.*, vol. 30, no. 3, 2020, Art. no. e2081, doi: 10.1002/nem.2081.

[11] T. Böttger, F. Cuadrado, G. Tyson, I. Castro, and S. Uhlig, "Open connect everywhere: A glimpse at the Internet ecosystem through the lens of the NetFlix CDN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 48, no. 1, pp. 28–34, 2018.

[12] Z. L. Zhang, "Feel free to cache: Towards an open CDN architecture for cloud-based content distribution," in *Proc. IEEE Int. Conf. Collaboration Technol. Syst. (CTS)*, 2014, pp. 488–490.

[13] J. S. Otto, M. A. Sánchez, J. P. Rula, T. Stein, and F. E. Bustamante, "namehelp: Intelligent client-side DNS resolution," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 287–288, 2012.

[14] R. Torres, A. Finamore, J. R. Kim, M. Mellia, M. M. Munafo, and S. Rao, "Dissecting video server selection strategies in the YouTube CDN," in *Proc. IEEE 31st Int. Conf. Distrib. Comput. Syst.*, 2011, pp. 248–257.

[15] U. Goel, M. P. Wittie, and M. Steiner, "Faster Web through client-assisted CDN server selection," in *Proc. IEEE 24th Int. Conf. Comput. Commun. Netw. (ICCCN)*, 2015, pp. 1–10.

[16] A. Biliris *et al.*, "CDN brokering," *Comput. Commun.*, vol. 25, no. 4, pp. 393–402, 2002.

[17] European Broadcasting Union (EBU). (2016). *EBU Flow Multi-CDN*. [Online]. Available: https://tech.ebu.ch/docs/events/IBC16/IBC%20Demo%20Fact%20Sheet_EBUFlow.pdf

[18] Citrix. *Intelligent Traffic Management*. Accessed: Sep. 18, 2020. [Online]. Available: https://www.citrix.com/products/citrix-intelligent-traffic-management/

[19] Lightflow. *LightFlow WisePath*. Accessed: Sep. 18, 2020. [Online]. Available: https://lightflow.media/lightflow-wisepath/

[20] *Developing Software for Multi-Access Edge Computing*. Accessed: Sep. 18, 2020. [Online]. Available: http://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp20_MEC_SoftwareDevelopment_FINAL.pdf

[21] R. Viola, A. Martin, M. Zorrilla, and J. Montalban, "MEC proxy for efficient cache and reliable multi-CDN video distribution," in *Proc. IEEE Conf. Broadband Multimedia Syst. Broadcast. (BMSB)*, Jun. 2018, pp. 1–7, doi: 10.1109/BMSB.2018.8436904.

[22] G. Carrozzo, F. Moscatelli, G. Solsona, O. P. Gordo, M. Keltsch, and M. Schmalohr, "Virtual CDNs over 5G networks: Scenarios and requirements for ultra-high definition media distribution," in *Proc. IEEE Int. Symp. Broadband Multimedia Syst. Broadcast. (BMSB)*, 2018, pp. 1–5.

[23] B. Frank *et al.*, "Pushing CDN-ISP collaboration to the limit," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 3, pp. 34–44, 2013.

[24] M. Wichtlhuber, R. Reinecke, and D. Hausheer, "An SDN-based CDN/ISP collaboration architecture for managing high-volume flows," *IEEE Trans. Netw. Service Manag.*, vol. 12, no. 1, pp. 48–60, Feb. 2015.

[25] S. Altamimi and S. Shirmohammadi, "QoE-fair DASH video streaming using server-side reinforcement learning," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 16, no 2s, p. 68, 2020, doi: 10.1145/3397227.

[26] F. Giannone, P. A. Frangoudis, A. Ksentini, and L. Valcarenghi, "Orchestrating heterogeneous MEC-based applications for connected vehicles," *Comput. Netw.*, vol. 180, Oct. 2020, Art. no. 107402, doi: 10.1016/j.comnet.2020.107402.

[27] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using ARIMA model and its impact on cloud applications—QoS," *IEEE Trans. Cloud Comput.*, vol. 3, no. 4, pp. 449–458, Dec. 2014.

[28] X. Dong, W. Fan, and J. Gu, "Predicting LTE throughput using traffic time series," *ZTE Commun.*, vol. 13, no. 4, pp. 61–64, 2015.

[29] A. Amin, L. Grunske, and A. Colman, "An automated approach to forecasting QoS attributes based on linear and non-linear time series modelling," in *Proc. 27th IEEE/ACM Int. Conf. Autom. Softw. Eng.*, 2012, pp. 130–139.

[30] A. Amin, A. Colman, and L. Grunske, "An approach to forecasting QoS attributes of Web services based on ARIMA and GARCH models," in *Proc. IEEE 19th Int. Conf. Web Services*, 2012, pp. 74–81.

[31] M. H. Zadeh and M. A. Seyyedi, "Qos monitoring for Web services by time series forecasting," in *Proc. 3rd Int. Conf. Comput. Sci. Inf. Technol.*, vol. 5, 2010, pp. 659–663.

[32] S. Belhaj and M. Tagina, "Modelling and prediction of the Internet end-to-end delay using recurrent neural networks," *J. Netw.*, vol. 4, no. 6, pp. 528–535, 2009.

[33] H. D. Trinh, L. Giupponi, and P. Dini, "Mobile traffic prediction from raw data using LSTM networks," in *Proc. IEEE 29th Annu. Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*, 2018, pp. 1827–1832.

[34] J. Wang *et al.*, "Spatiotemporal modelling and prediction in cellular networks: A big data enabled deep learning approach," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, 2017, hboxpp. 1–9.

[35] A. Azari, P. Papapetrou, S. Denic, and G. Peters, "Cellular traffic prediction and classification: A comparative evaluation of LSTM and ARIMA," 2019. [Online]. Available: arXiv:1906.00939.

[36] A. Azari, P. Papapetrou, S. Denic, and G. Peters, "User traffic prediction for proactive resource management: Learning-powered approaches," 2019. [Online]. Available: arXiv:1906.00951.

[37] *Information Technology—MPEG Systems Technologies—Part 7: Common Encryption in ISO Base Media File Format Files*, ISO/IEC Standard 23001-7, 2016.

[38] T. Lohmar, T. Einarsson, P. Fräjdh, F. Gabin, and M. Kampmann, "Dynamic adaptive HTTP streaming of live content," in *Proc. IEEE Int. Symp. World Wireless Mobile Multimedia Netw.*, 2011, pp. 1–8.

[39] E. Thomas, M. O. Van Deventer, T. Stockhammer, A. C. Begen, and J. Famaey, "Enhancing MPEG DASH performance via server and network assistance," in *Proc. IBC Conf.*, 2015.

[40] L. Nussbaum, "An overview of Fed4FIRE testbeds and beyond?" in *Proc. GEFI-Global Exp. Future Internet Workshop*, 2019, p. 9.

[41] N. Makris, C. Zarafetas, S. Kechagias, T. Korakis, I. Seskar, and L. Tassiulas, "Enabling open access to LTE network components; the NITOS testbed paradigm," in *Proc. 1st IEEE Conf. Netw. Softw. (NetSoft)*, Apr. 2015, pp. 1–6.

[42] S. Lederer, C. Mueller, C. Timmerer, C. Concolato, J. Le Feuvre, and K. Fliegel, "Distributed DASH dataset," in *Proc. 4th ACM Multimedia Syst. Conf.*, 2013, pp. 131–135, doi: 10.1145/2483977.2483994.

[43] *GStreamer: Open Source Multimedia Framework*. Accessed: Sep. 18, 2020. [Online]. Available: https://gstreamer.freedesktop.org/

[44] *Node.js: Asynchronous Event Driven JavaScript Runtime*. Accessed: Sep. 18, 2020. [Online]. Available: https://nodejs.org/en/

[45] *TensorFlow: End-to-End Open Source Platform for Machine Learning*. Accessed: Sep. 18, 2020. [Online]. Available: https://www.tensorflow.org/

[46] (2017). *Cisco Inc: Visual Networking Index: Forecast and Methodology, 2017–2022*. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11–741490.html

[47] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng, "Understanding user behavior in large-scale video-on-demand systems," *ACM SIGOPS Oper. Syst. Rev.*, vol. 40, no. 4, pp. 333–344, 2006.

[48] C. Gormley and Z. Tong, *Elasticsearch: The Definitive Guide: A Distributed Real-Time Search and Analytics Engine*. London, U.K.: O'Reilly Media, 2015.

[49] Y. Gupta, *Kibana Essentials*. New York, NY, USA: Packt, 2015.

[50] S. Verbrugge *et al.*, "Methodology and input availability parameters for calculating OpEx and CapEx costs for realistic network scenarios," *J. Opt. Netw.*, vol. 5, no. 6, pp. 509–520, 2006.

[51] DaCast. (2019). *2019 Live Streaming CDN Pricing Comparison*. [Online]. Available: https://www.dacast.com/blog/blog-live-streaming-cdn-pricing/

[52] CDNPerf. *CDN Performance Analytics & Comparison*. Accessed: Sep. 18, 2020. [Online]. Available: https://www.cdnperf.com/

[53] Wowza Media Systems. *Wowza Streaming Cloud Plans*. Accessed: Sep. 18, 2020. [Online]. Available: https://www.wowza.com/pricing/streaming-cloud-plans

Roberto Viola received the Computer and Telecommunication Engineering degree and the Advanced degree in telecommunication engineering from the University of Cassino and Southern Lazio, Italy, in 2014 and 2016, respectively. He is currently pursing the Ph.D. degree on video content distribution in 5G networks with the University of the Basque Country (UPV/EHU). He is with the Department of Digital Media, Vicomtech, where he is a Research Associate involved in projects dealing with multimedia services and network infrastructure.

Angel Martin received the Engineering degree from University Carlos III in 2003, and the Ph.D. degree from UPV/EHU in 2018. He is with the Department of Digital Media, Vicomtech, where he is working with multimedia services and 5G infrastructures projects. He developed in Prodys an standard MPEG-4 AVC/H.264 codec for DSP (2003–2005). He worked in media streaming and encoding research (2005–2008) with Telefonica. He worked in the fields of smart environments and ubiquitous and pervasive computing (2008–2010) with Innovalia.

Javier Morgade (Senior Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in telecommunications engineering from the UPV/EHU, Spain, in 2006, 2009, and 2014, respectively. He is with Digital Media Department, Vicomtech, Spain. After finalizing the Ph.D. degree at UPV/EHU, he joined Samsung Electronics Research and Development U.K., where he was involved in ATSC 3.0 standardization. In 2014, he joined Fraunhofer IIS, Germany, where he was engaged in 3GPP RAN/SA standardization. Activity he continued in 2017 for German Public Broadcasters when he joined the IRT, Munich. Before joining Vicomtech, he was employed by Smart Mobile Labs AG, Munich, working on several aspects of NFV, Cloud/Edge Computing (SCaaS, IaaS, PaaS, and SaaS) for E2E content distribution in 4G/5G networks in 2018. He has authored over 40 scientific publications and patents having participated in a wide list of public funded projects (last ones: CDN-X-ALL, 5G-MOBIX, LIPS, 5G-TODAY, 5G-ESSENCE, 5G-XCast, and IMB5) and projects for the European Space Agency. He is an active member of the OpenAirInterface5G Software Alliance.

Stefano Masneri received the M.Sc. degree in telecommunications from Università degli studi di Brescia, Italy, in 2008. He is with the Department of Digital Media, Vicomtech. He has previously been working as a Research Associate with CNIT, Brescia, and Fraunhofer HHI, Berlin, participating in several national and European projects. After that, he worked as a Scientific Software Developer with the Max Planck Institute for Brain Research and with AGT International as a Senior Data Scientist. He focuses his research in machine learning, computer vision, and interactive technologies.

Mikel Zorrilla received the Telecommunication Engineering degree from Mondragon Unibertsitatea, in 2007, and the Advanced degree and the Ph.D. degree in computer science from UPV/EHU in 2012 and 2016, respectively. He is the Head of the Digital Media Department, Vicomtech. He has participated in many international research projects, such as MediaScape or Hbb4All European Projects. Previously he has held positions with IK4-Ikerlan as an Assistant Researcher (2002–2006) and with Deusto Business School as an Associate Professor in media in 2014.

Pablo Angueira (Senior Member, IEEE) received the M.S. and Ph.D. degrees in telecommunications engineering from UPV/EHU in 1997 and 2002, respectively. He joined the Communications Engineering Department, UPV/EHU in 1998, where he is a Full Professor and part of the TSR Staff (Signal Processing and Radiocommunications) Research Group, involved in digital broadcasting technologies with contributions to the ITU-R WG6 and WG3. His main research interests are network planning and spectrum management for digital terrestrial broadcast technologies, and broadcasting in 5G networks.

Jon Montalban (Senior Member, IEEE) received the M.S. and Ph.D. degrees in telecommunications engineering from UPV/EHU in 2009 and 2014, respectively. Since 2009, he has been a part of the TSR (Signal Processing and Radiocommunications) Research Group, UPV/EHU, where he is a Postdoctoral Researcher involved in digital terrestrial TV broadcasting projects. His current research interests include digital communications and digital signal processing for mobile reception of broadband wireless communications systems in 5G.